TIE-0250x concurrency mandatory projects
Project 2: C++ threads and mutual exclusion, spring 2018

2018-03-26: course code has comment clarifications.
`git pull upstream master` will sync your repository
2018-02-19: published

This document contains description for one project, remember also to read the general information about the course projects.

PLEASE NOTE: course library code will NOT be visible in your course-gitlab -repository until you have pulled it from the course master repository! (see general project information)

---

1. Problem description

The main points in this project are:

- Create a C++ program using multiple threads of execution
- Analyse problem/program for race conditions and solve them using locks

2. Image transformation library

Course provides a Linux library for making modifications to image files. Functionality is provided by an API (documented in a C++ header file), which defines the available operations.

API definition

Implementation for this API is located in directory `concur2018lib`. This assumes that:

1. You are running in a Linux environment (e.g. linux-desktop.cc.tut.fi)
2. ImageMagick software is installed (requires `convert` command).
3. You are compiling and running your program with QtCreator.

   All these should work "out of the box" when using the TUT environment `linux-desktop.cc.tut.fi`.

3. Functionality and requirements

- Your submitted code MUST start several (lets say 4-8) threads to run image conversions (e.g. start the same functionality as present in `main.cpp: convertFiles` multiple times).
- Code is implemented using C++ threads and synchronisation methods (no Qt threads nor any external concurrency libraries).
- Only features documented in the API can be relied on. Anything else is undefined behaviour.
- DOCUMENT (in the code or with a separate PDF-document) ALL the concurrency problems you identified and solved during making your code.

Submission

Commit your SOURCE code (no executables nor QtCreator build-files please) to `course-gitlab.tut.fi`. General submission rules define how you submit your work from GIT to the Repolainen submission system.

4. Problems?

Send any questions about the project to the course email: rinn@tut.fi